Guia de usuario

ICARO

1.0



Valentin Basel

Guia de usuario ICARO
1.0
Edición 0

Autor Valentin Basel valentinbasel@gmail.com

Preface

1. Convenciones del Documento

Este manual utiliza varias convenciones para resaltar algunas palabras y frases y llamar la atención sobre ciertas partes específicas de información.

En ediciones PDF y de papel, este manual utiliza tipos de letra procedentes de *Liberation Fonts*[1]. Liberation Fonts también se utilizan en ediciones de HTML si están instalados en su sistema. Si no, se muestran tipografías alternativas pero equivalentes. Nota: Red Hat Enterprise Linux 5 y siguientes incluyen Liberation Fonts predeterminadas.

1.1. Convenciones Tipográficas

Se utilizan cuatro convenciones tipográficas para llamar la atención sobre palabras o frases específicas. Dichas convenciones y las circunstancias en que se aplican son las siguientes:

`Negrita monoespaciado`

Utilizada para resaltar la entrada del sistema, incluyendo comandos de shell, nombres de archivo y rutas. También se utiliza para resaltar teclas claves y combinaciones de teclas. Por ejemplo:

> Para ver el contenido del archivo **`my_next_bestselling_novel`** en su directorio actual de trabajo, escriba el comando **`cat my_next_bestselling_novel`** en el intérprete de comandos de shell y pulse **`Enter`** para ejecutar el comando.

El ejemplo anterior incluye un nombre de archivo, un comando de shell y una tecla clave. Todo se presenta en negrita-monoespaciado y distinguible gracias al contexto.

Las combinaciones de teclas se pueden distinguir de las teclas claves mediante el guión que conecta cada parte de una combinación de tecla. Por ejemplo:

> Pulse **`Enter`** para ejecutar el comando.
>
> Pulse **`Control`**+**`Alt`**+**`F2`** para cambiar a la primera terminal virtual. Pulse **`Control`**+**`Alt`**+**`F1`** para volver a su sesión de Ventanas-X.

La primera oración resalta la tecla clave determinada que se debe pulsar. La segunda resalta dos conjuntos de tres teclas claves que deben ser presionadas simultáneamente.

Si se discute el código fuente, los nombres de las clase, los métodos, las funciones, los nombres de variables y valores de retorno mencionados dentro de un párrafo serán presentados en **`Negrita-monoespaciado`**. Por ejemplo:

> Las clases de archivo relacionadas incluyen **`filename`** para sistema de archivos, **`file`** para archivos y **`dir`** para directorios. Cada clase tiene su propio conjunto asociado de permisos.

Negrita proporcional

Esta denota palabras o frases encontradas en un sistema, incluyendo nombres de aplicación, texto de cuadro de diálogo, botones etiquetados, etiquetas de cajilla de verificación y botón de radio; títulos de menú y títulos del sub-menú. Por ejemplo:

[1] https://fedorahosted.org/liberation-fonts/

> Seleccionar **Sistema** → **Preferencias** → **Ratón** desde la barra del menú principal para lanzar **Preferencias de Ratón**. En la pestaña de **Botones**, haga clic en la cajilla **ratón de mano izquierda** y luego haga clic en **Cerrar** para cambiar el botón principal del ratón de la izquierda a la derecha (adecuando el ratón para la mano izquierda).
>
> Para insertar un caracter especial en un archivo de **gedit**, seleccione desde la barra del menú principal **Aplicaciones** → **Accessories** → **Mapa de caracteres**. Luego, desde la barra de menúes de **mapa de caracteres** elija **Búsqueda** → **Hallar…**, teclee el nombre del caracter en el campo **Búsqueda** y haga clic en **Siguiente**. El caracter buscado se resaltará en la **Tabla de caracteres**. Haga doble clic en este caracter resaltado para colocarlo en el campo de **Texto para copiar** y luego haga clic en el botón de **Copiar**. Ahora regrese a su documento y elija **Editar** → **Pegar** desde la barra de menú de **gedit**.

El texto anterior incluye nombres de aplicación; nombres y elementos del menú de todo el sistema; nombres de menú de aplicaciones específicas y botones y texto hallados dentro de una interfaz gráfica de usuario, todos presentados en negrita proporcional y distinguibles por contexto.

`Itálicas-negrita monoespaciado` o ***`Itálicas-negrita proporcional`***

Ya sea negrita monoespaciado o negrita proporcional, la adición de itálicas indica texto reemplazable o variable. Las itálicas denotan texto que usted no escribe literalmente o texto mostrado que cambia dependiendo de la circunstancia. Por ejemplo:

> Para conectar a una máquina remota utilizando ssh, teclee **`ssh`** ***`nombredeusuario@dominio.nombre`*** en un intérprete de comandos de shell. Si la máquina remota es **`example.com`** y su nombre de usuario en esa máquina es john, teclee **`ssh john@example.com`**.
>
> El comando **`mount -o remount`** ***`file-system`*** remonta el sistema de archivo llamado. Por ejemplo, para volver a montar el sistema de archivo **`/home`**, el comando es **`mount -o remount /home`**.
>
> Para ver la versión de un paquete actualmente instalado, utilice el comando **`rpm -q`** ***`paquete`***. Éste entregará el resultado siguiente: ***`paquete-versión-lanzamiento`***.

Observe las palabras en itálicas y negrita sobre — nombre de usuario, domain.name, sistema de archivo, paquete, versión y lanzamiento. Cada palabra es un marcador de posición, tanto para el texto que usted escriba al ejecutar un comando como para el texto mostrado por el sistema.

Aparte del uso estándar para presentar el título de un trabajo, las itálicas denotan el primer uso de un término nuevo e importante. Por ejemplo:

> Publican es un sistema de publicación de *DocBook*.

1.2. Convenciones del documento

Los mensajes de salida de la terminal o fragmentos de código fuente se distinguen visualmente del texto circundante.

Los mensajes de salida enviados a una terminal se muestran en **`romano monoespaciado`** y se presentan así:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Los listados de código fuente también se muestran en **`romano monoespaciado`**, pero se presentan y resaltan de la siguiente manera:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

1.3. Notas y Advertencias

Finalmente, utilizamos tres estilos visuales para llamar la atención sobre la información que de otro modo se podría pasar por alto.



Nota

Una nota es una sugerencia, atajo o enfoque alternativo para una tarea determinada. Ignorar una nota no debería tener consecuencias negativas, pero podría perderse de algunos trucos que pueden facilitarle las cosas.



Importante

Los cuadros con el título de importante dan detalles de cosas que se pueden pasar por alto fácilmente: cambios de configuración únicamente aplicables a la sesión actual, o servicios que necesitan reiniciarse antes de que se aplique una actualización. Ignorar estos cuadros no ocasionará pérdida de datos, pero puede causar enfado y frustración.



Advertencia

Las advertencias no deben ignorarse. Ignorarlas muy probablemente ocasionará pérdida de datos.

2. ¡Necesitamos sus comentarios!

Debe sobrescribir este archivo de comentarios creando su propio archivo local Feedback.xml.

ICARO

1.1. introducción

test test

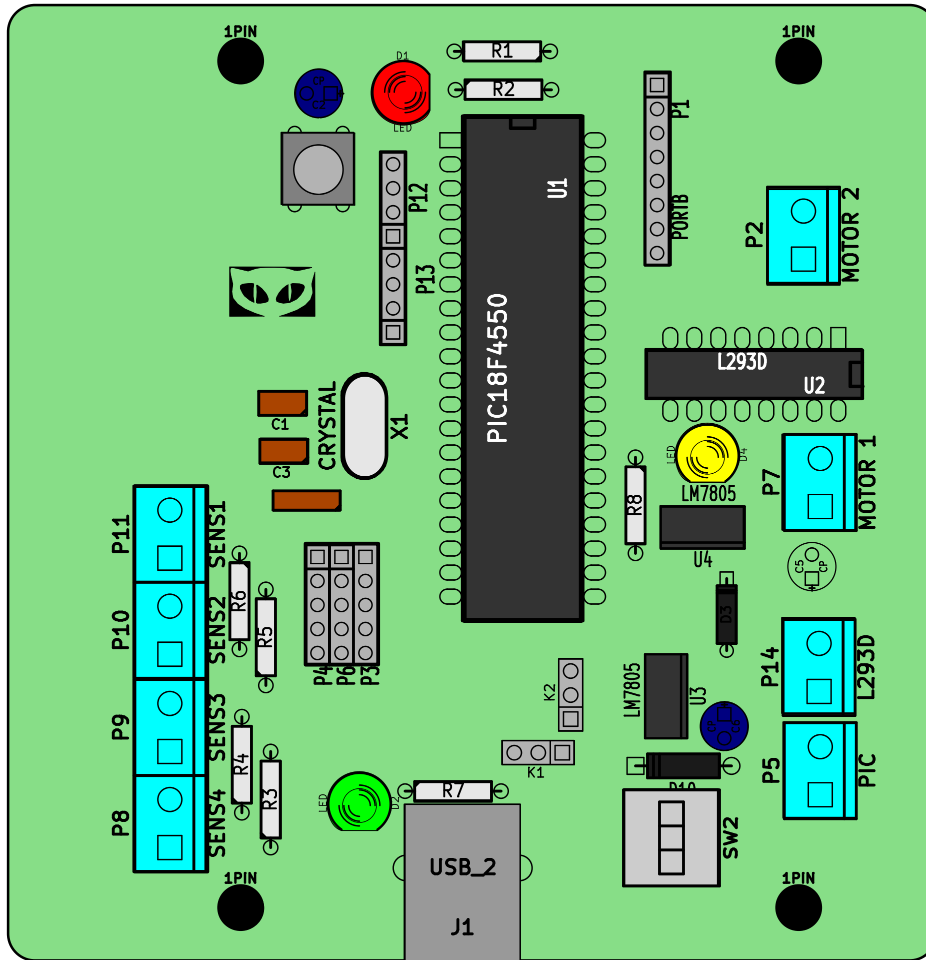
1.2. Instalación manual de `ICARO`.

Para instalar manualmente **`ICARO`** hay que seguir los siguientes pasos (como **`root`**):

1. #> yum -y install pygame pywebkitgtk pygtksourceview
2. #> yum -y install sdcc gputils
3. copiar el archivo pic18f4550.h en la carpeta sdcc-include a **`/usr/share/sdcc/include/pic16/`**
4. copiar el archivo pic18f2455.h en la carpeta sdcc-include a **`/usr/share/sdcc/include/pic16/`**
5. copiar el archivo libdev18f4550.lib en la carpeta sdcc-include a **`/usr/share/sdcc/lib/pic16/`**
6. copiar el archivo macro.h en la carpeta sdcc-include a **`/usr/share/sdcc/include/pic16/`**
7. En algunos sistemas linux puede suceder que necesitemos pertenecer al grupo "admin" para poder acceder al dispositivo USB que representa la placa.

Placa NP05

La placa ICARO NP05 está diseñada para trabajar nativamente con el puerto USB de su computadora. Por defecto, toma la alimentación del puerto USB para encender el microcontrolador y arrancar el sistema.



Placa NP05 .



Nota

Con el selector K1 se puede seleccionar la alimentación directa de USB o de una fuente externa.

El primer paso es conectar el cable USB a la placa y encenderla apretando el boton SW2. El LED verde (D2) se encenderá indicando que hay tensión en el circuito que va al microcontrolador. Luego de 3 segundos, se encendera el LED rojo (D1) indicando que el PIC esta operativo.

El LED rojo inidica que el pic se inició correctamente y está ejecutando el código cargado previamente; de esa forma se puede saber si el sistema tiene tensión y está operativo.

LED indicador de funcionamiento

LED indicador de tension

Boton de encendido

Conectar al PC

Esquema de conexión inicial

Con la placa ya conectada y encendida, podemos ver si la pc reconoce al microcontrolador. Para eso usamos el comando **lsusb**, el cual debería mostrar en la salida el id de la placa (entre otros).

```
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

En principio los sitemas GNU/linux bloquean al usuario normal para poder trabajar con los puertos USB directamente. Para tener los privilegios de usuario y poder mandar datos por el puerto USB a nuestra placa hay varias formas, la más sencilla es ser usuario **root** y trabajar desde ahí.



Advertencia

Es muy arriesgado trabajar como root, lo ideal es dar permisos de escritura en el puerto mediante **udev** como se explicará más adelante.



Nota

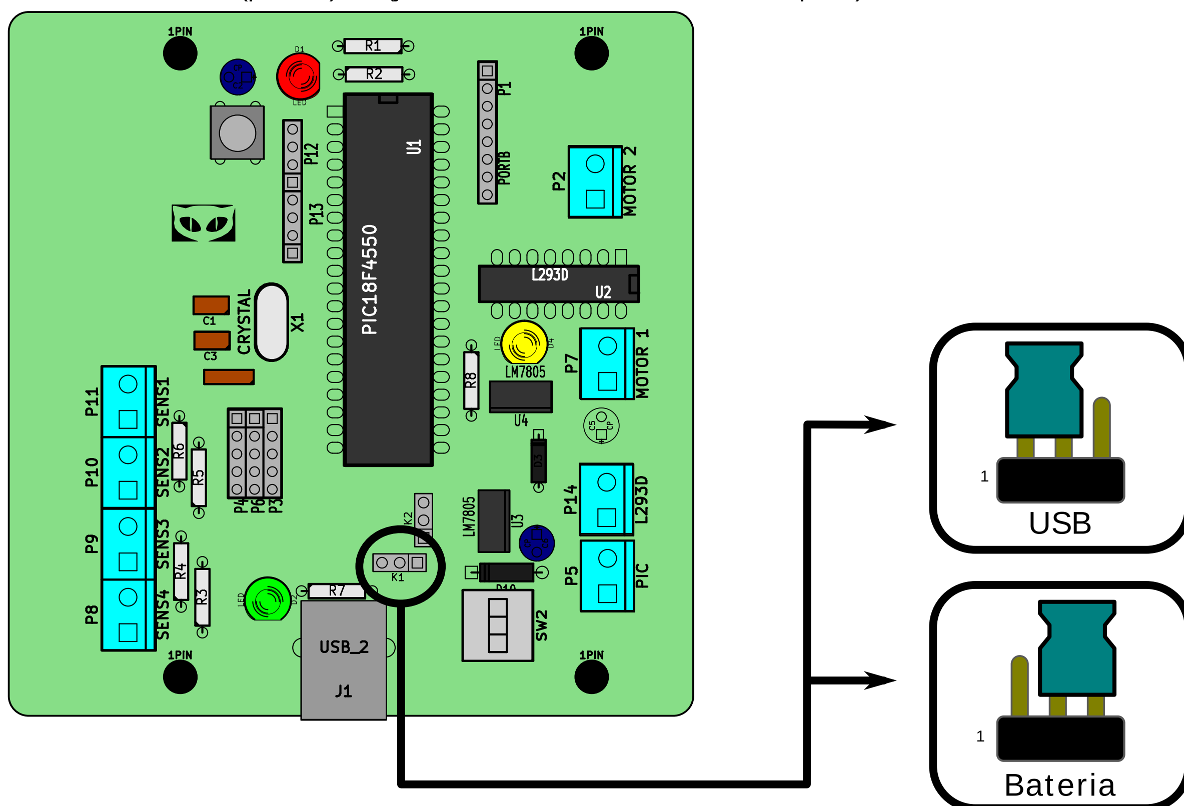
Las placas ICARO NP05 trabajan basadas en el bootlader del proyecto Pinguino, por lo tanto todo el desarrollo para este proyecto es aplicable a las placas NP05 .

Una vez que el sistema está funcionando, podemos comenzar a cargar el firmware que vamos a usar. El firmware es código C++ compilado con **`SDCC`** que se puede subir a la placa mediante el programa **`docker`**, el cual se puede conseguir del proyecto VASCO PUF.

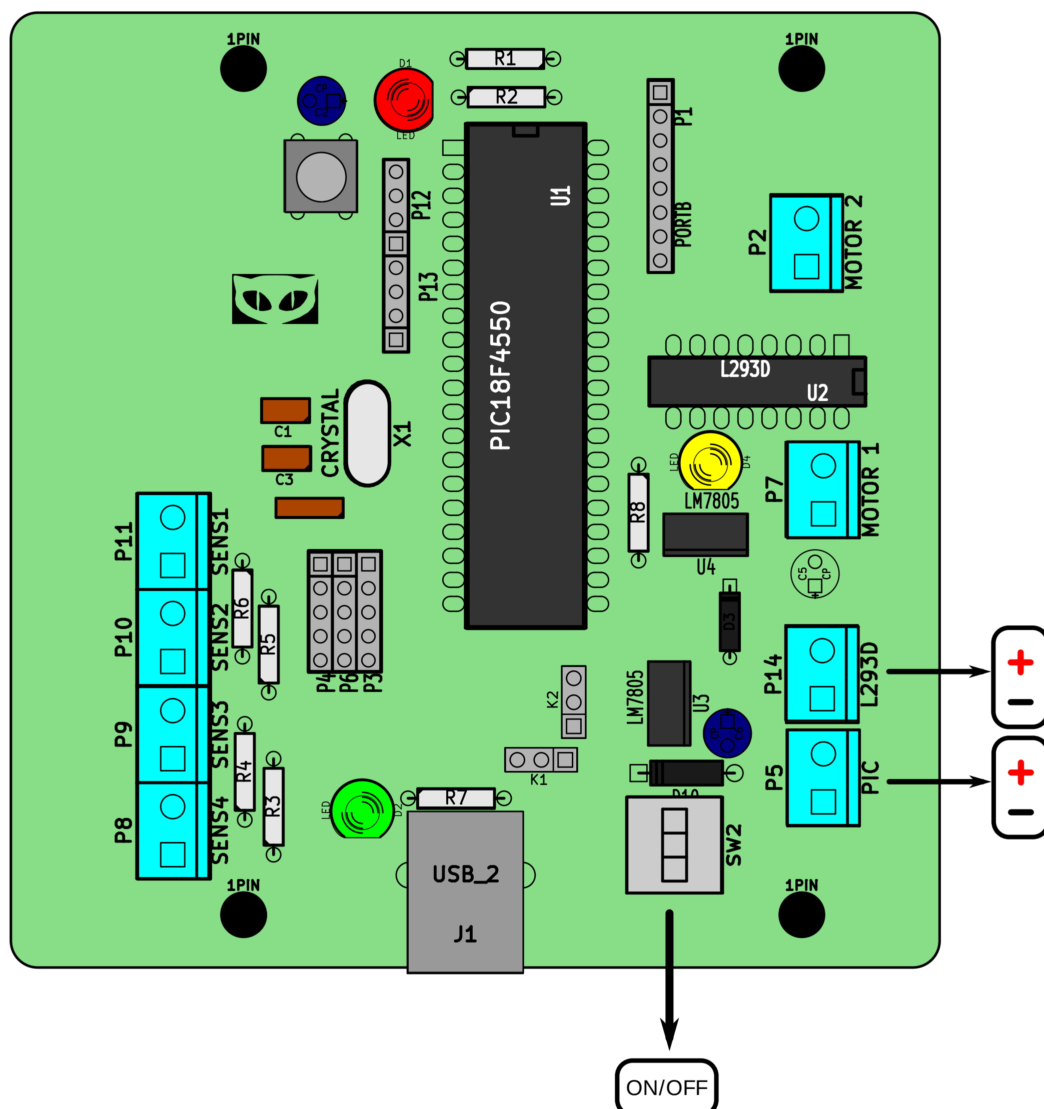
Sin embargo, las placas ICARO NP05 estan preparadas para trabajar de 2 formas: con un firmware estándar previamente cargado para funcionar conectado al puerto usb y usar el programa **`TurtleArt`**, o con un firmware a medida diseñado con el programa **`icaro-bloques 1.0`**

2.1. Esquema eléctrico de la placa NP05

El esquema de conexión eléctrico de las placas NP05 está separado en dos: por un lado la alimentación del PIC (que puede ser directamente desde USB o baterías) y la conexión para el driver L293D (que usa baterías directamente). Para poder seleccionar el voltaje USB o de pilas (para el PIC) se usa el JUMPER (puente) K1 (justo arriba del conector USB de tipo D).



Las borneras P5 y P14 son para conectar las baterías, tanto del pic como para el driver de potecia para motores CC. Es importante ver las conexiones de los polos de las baterías, en el esquema siguiente se puede apreciar cómo deben ir conectados los polos positivos y negativos de las pilas que usemos con la placa.



2.2. Habilitación de escritura en el puerto

Cuando se conecta por primera vez una placa NP05 a una computadora, por defecto los sistemas GNU/linux no habilitan el modo escritura en el puerto, para ello tenemos que habilitarlo manualmente.

La forma más sencilla es la dicha más arriba, que es trabajar como **`root`**, sin embargo esta práctica no es recomendada. Una de las formas es siendo **`root`** y dar permisos de lectura y escritura al puerto **`/dev/ttyACM0`**.

`chmod 777` ***`/det/ttyACM0`***



Nota

El valor de ACM, varía en función de los puertos que hay conectados, puede ser /dev/ttyACM0, 1, 2, 3 etc.

Sin embargo, cada vez que se desconecte la placa de la computadora, los privilegios de usuario se borrarán y habrá que aplicarlos de vuelta.

Para poder trabajar con el puerto ttyACM hay que agregar nuestro usuario al grupo **`dialout`**

#> usermod -a -G dialout $USER



Nota

En realidad icaro-bloques 1.0 no necesita específicamente tener permisos de escritura en el puerto /dev/ttyACM. ACM es para poder comunicar con el plugin Tortcaro o para usar Python con el modulo pyserial directamente,

2.3. Creación de reglas udev

`udev` es el gestor de dispositivos que usa el kernel Linux en su versión 2.6. Su función es controlar los ficheros de dispositivo en /dev. Es el sucesor de devfs y de hotplug, lo que significa que maneja el directorio /dev y todas las acciones del espacio de usuario al agregar o quitar dispositivos, incluyendo la carga de firmwares.

Mediante **`udev`**, podemos darles permisos de escritura para un grupo a nuestro hardware, y evitar la necesidad de se **`root`** con los posibles problemas de seguridad que eso implica.

Para crear las reglas **`udev`** necesarias para poder trabajar con la placa NP05 , tenemos que copiar los 2 archivos .udev de la carpeta **`udev`** donde tenemos instalado **`icaro-bloques 1.0`** a la dirección **`/etc/udev/rules.d/`**.

#> su -c "cp *.udev /etc/udev/rules.d"

#> groupadd microchip

#> usermod -a -G microchip $USER



Nota

Para poder copiar lor archivos udev a **`/etc/udev/rules.d/`** hay que tener privilegios **`root`**. Pero solo se usa una única vez.

2.4. Lista de componentes

La mayoría de los compontentes de la placa NP05 son de fácil adquisición, generalmente lo más difícil de conseguir son el pic y el driver de potencia L293B.

Tabla 2.1. Listado de componentes placa NP05

cantidad	Descripcion
1	PIC 18f4550
1	L293D (o L293B)
2	Diodos 1n4007
2	LM7805
1	Zócalo 2x8 pines
1	Zócalo 2x20 pines

cantidad	Descripcion
1	Tira de pines (40 pines)
3	Leds 5 mm de distintos colores (rojo , verde ,amarillo)
1	Conector USB hembra tipo B
1	Cristal de 20 MHz
2	Capacitores ceramicos 22 PF
1	capacitor ceramico 220 nF
1	Capacitor electrolítico 10 uFx 16V
2	Capacitor electrolítico 100 uFx 16V
1	Pushbutton para pcb
1	Botón de encendido para pcb (botón interruptor)
8	Borneras
10	Resistencias 10K Ohms
10	Resistencias 470 Ohms

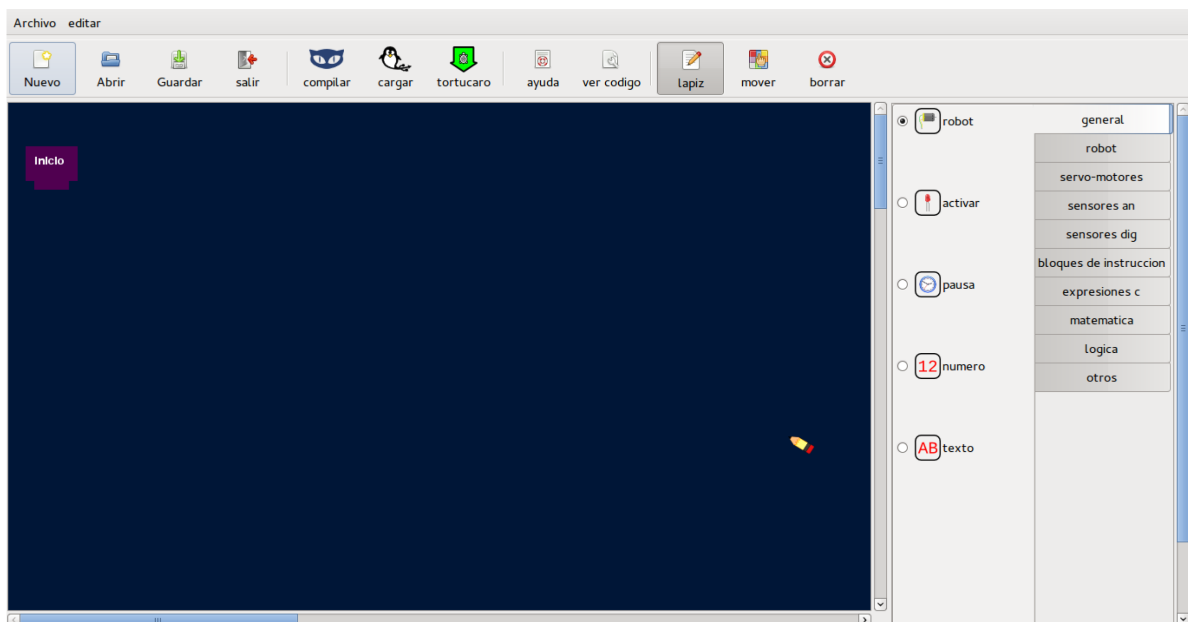
Manual de usuario icaro-bloques 1.0

3.1. `icaro-bloques 1.0`

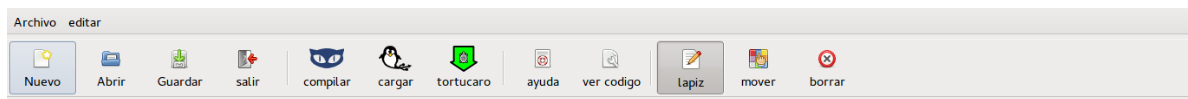
`icaro-bloques 1.0` es un programa para generar y cargar firmware en los pics 18f4550 con bootloaders, VASCO-PUF (del proyecto PINGUINO). Su funcionamiento se basa en "apilar" bloques que representan líneas de instrucciones ANSI C (SDCC), permitiendo generar código C de forma muy sencilla sin conociminetos de programación o electrónica.

3.2. Disposición de la ventana `icaro-bloques 1.0`

`icaro-bloques 1.0` tienen 3 áreas de trabajo: la barra de heramientas, la de componentes y la zona de trabajo.



3.3. Barra de herramientas



La barra de herramientas aloja los botones básicos para trabajar con **`icaro-bloques 1.0`**.

1. **`Nuevo`**: borra los bloques y deja solamente el bloque "inicio".

2. **`Abrir`**: muestra el selector de archivos para abrir un archivo .icr

3. **`Guardar`**: muestra el selector de archivos para guardar un archivo .icr

4. **`Salir`**: sale del programa.

5. **`Compilar`**: una vez montados los bloques, concatenados al bloque "inicio", hay que apretar el botón **`Compilar`** para poder generar el código fuente C.

6. **`Cargar`**: después de **`Compilar`** nuestro código y que no muestre error se puede proceder a usar el botón **`Cargar`** para subir el archivo como firmware a nuestra placa.

7. **`Tortucaro`**: carga un firmware específico para poder trabajar con apicaro, python y con el plugin Tortucaro para TurtleArt.

8. **`Ayuda`**: muestra este manual.

9. **`Ver código`**: muestra el código fuente C generado por el botón **`Compilar`**, sirve para revisar las lineas de código generadas.

10. **`Lápiz`**: dibuja los componentes seleccionados en la **`Barra de componentes`**.

11. **`Mover`**: mueve los componentes de la **`Zona de trabajo`**.

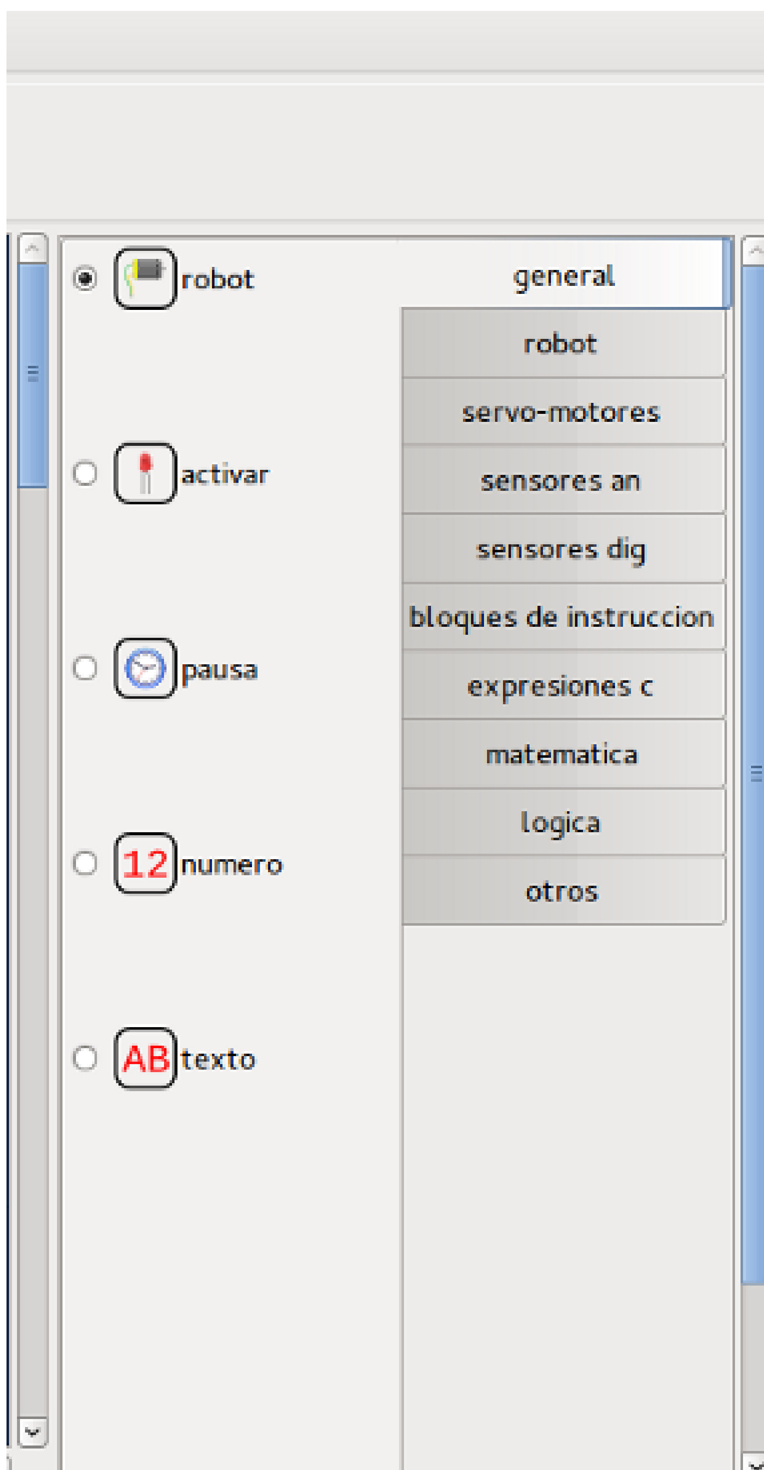
12. **`Borrar`**: borra los componentes de la **`Zona de trabajo`**



Nota

Siempre para poder crear un programa, primero hay que compilarlo con el botón **`Compilar`** y despues cargarlo con el botón **`cargar`**.

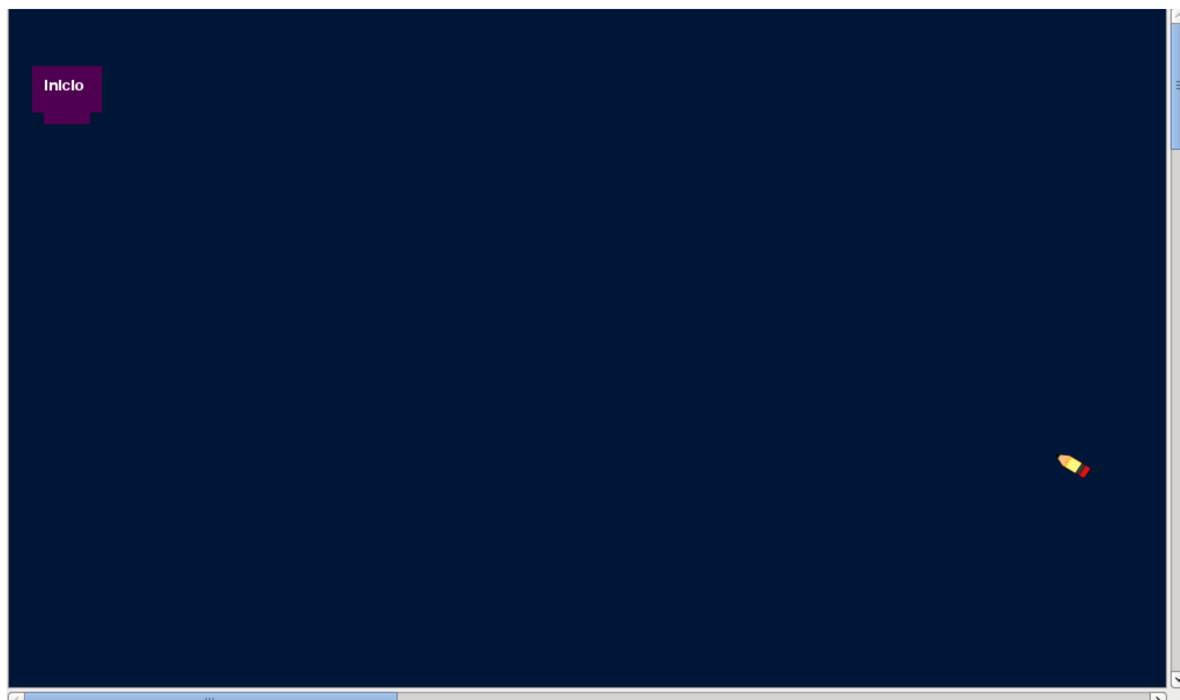
3.4. Barra de componente



La barra de componentes es donde están todos los bloques que se pueden acoplar entre sí.

Se separa en dos partes: a la derecha están los grupos (general, robot, matemática, etc etc) y a la izquierda están los componentes de cada grupo. Solo se puede seleccionar un componente a la vez de cada grupo.

3.5. Zona de trabajo



La zona de trabajo es donde van los bloques de componenentes que formarán el código fuente. Solo tiene un componente que es el bloque "inicio", todos los demás bloques deben estar unidos a este bloque en forma de cadena. "Inicio" es el primer bloque de la cadena de código y es el único bloque que no puede ser borrado.

3.6. Tipo de componentes de `icaro-bloques 1.0`

test test test test

3.7. Cómo crear un programa

test test test test

3.8. Compilar

test test test test

3.9. Cargar el firmware a la placa

test test test test

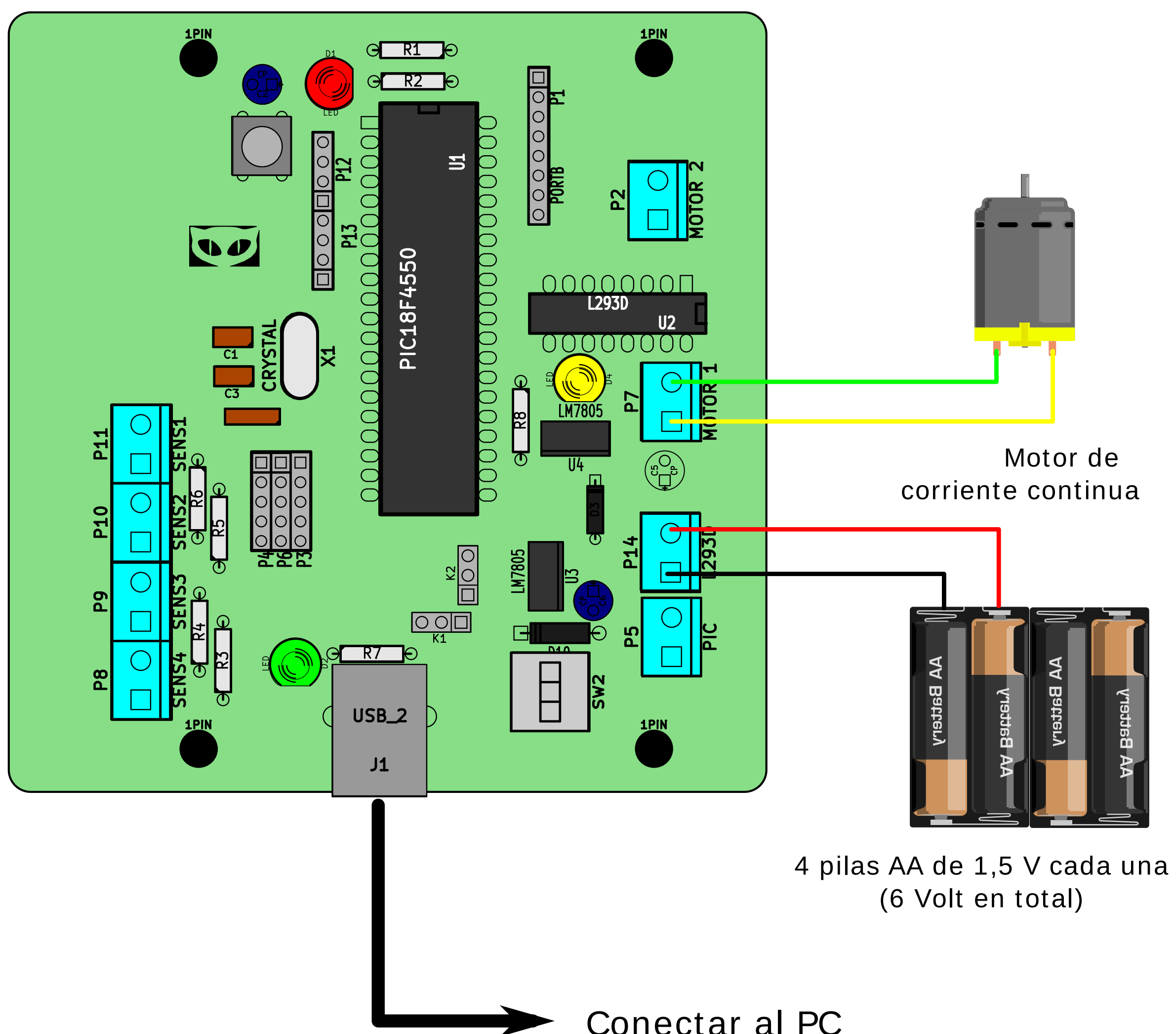
Ejemplos de programación

4.1. Primer programa

Vamos a hacer el primer test de la placa ICARO NP05 ; para ello necesitamos:

1. 4 pilas AA de 1.5 volt y un portapilas para las 4
2. 1 motor de corriente continua que trabaje a 5 volt (cualquier motor de juguete está bien)
3. 1 placa ICARO NP05 y cable USB tipo D
4. el software **`icaro-bloques 1.0`**

Conectar el motor y las pilas como se indca en el gráfico.



Los conectores P2 y P7 son para controlar motores de corriente continua (o un motor paso a paso del tipo unipolar). la placa ICARO NP05 trae incorporada un driver de potencia (puente H) de tipo L293D que puede manejar tensiones de hasta 36 volts. Sin embargo, en el esquema de la placa, la tensión se reduce a 5 volt para poder trabajar con motores de poca potencia (típicamente motores CC de juguetes).

Por seguridad el driver de potencia tiene una alimentación separada del resto de la placa, por eso necesita pilas (o una fuente de hasta 12 volt) para poder funcionar.

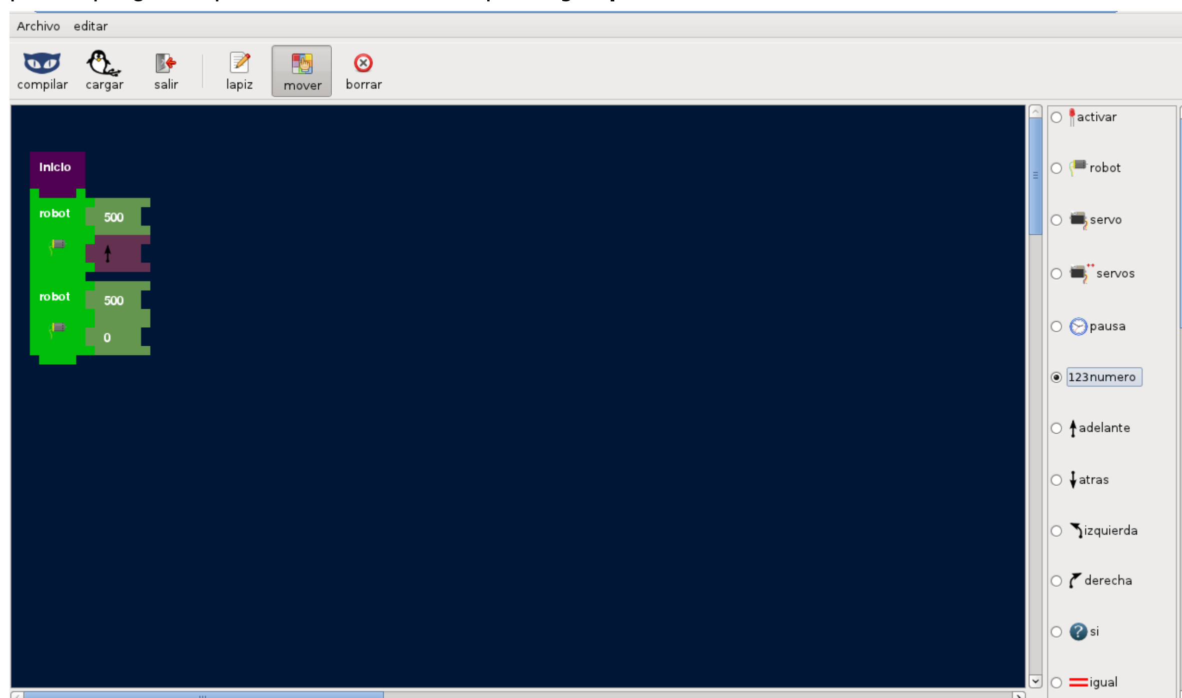
En cuanto conectamos las pilas al conector P14, el LED amarillo (d4) se encenderá, indicando que esa sección tiene tensión.



advertencia

Observar bien la disposición de los polos positivo y negativo en la placa; si bien hay diodos de protección por si nos equivocamos y ponemos los cables al revés, no es recomendable dejar mucho tiempo con la tensión invertida la placa. Si los LED de tensión (verde y amarillo) no encienden, revizar las conexiones.

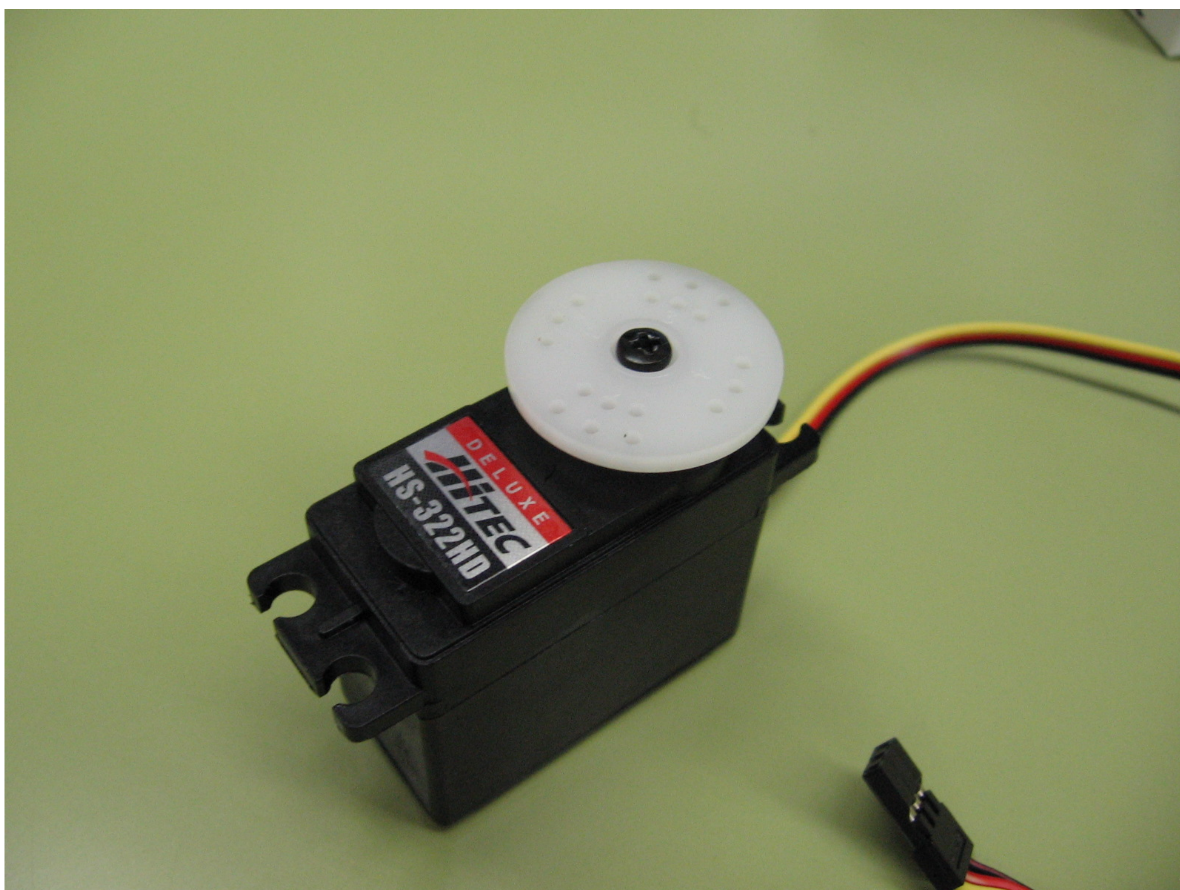
Con el cableado ya conectado abrimos el programa **`icaro-bloques 1.0`**, y cargamos nuestro primer programa para icaro desde la carpeta **`ejemplos/hola_mundo.icr2`**.



4.2. Servomotores

Un servomotor (también llamado "servo") es un dispositivo similar a un motor de corriente continua que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y mantenerse estable en dicha posición.

Los servomotores hacen uso de la modulación por ancho de pulsos (PWM) para controlar la dirección o posición de los motores de corriente continua. La mayoría trabaja en la frecuencia de los cincuenta hercios, así las señales PWM tendrán un periodo de veinte milisegundos. La electrónica dentro del servomotor responderá al ancho de la señal modulada. Si los circuitos dentro del servomotor reciben una señal de entre 0,5 a 1,4 milisegundos, este se moverá en sentido horario; entre 1,6 a 2 milisegundos moverá el servomotor en sentido antihorario; 1,5 milisegundos representa un estado neutro para los servomotores estándares.



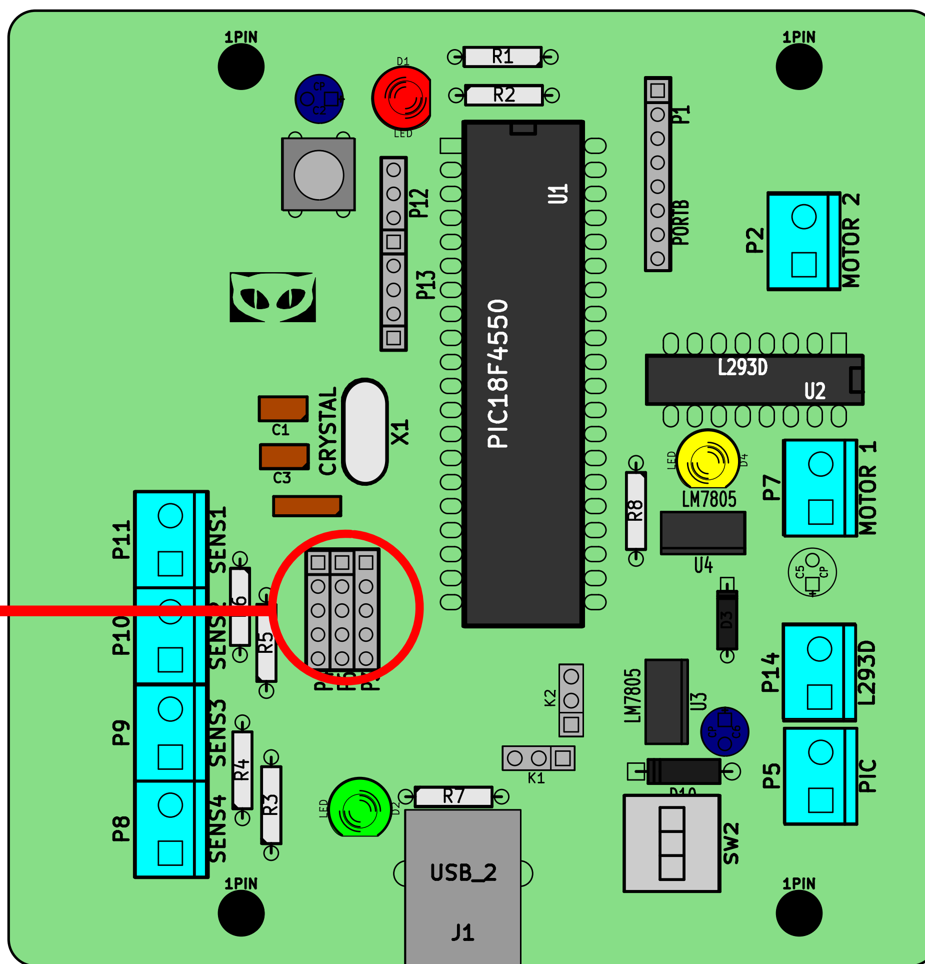
Servomotor de aeromodelismo.

La placa NP05 posee 5 conectores para servos, los cuales pueden ser controlados al mismo tiempo. Toman la tensión de alimentación de la misma fuente que el PIC, por lo tanto se pueden usar desde la alimentación de USB sin necesidad de baterías.



Advertencia

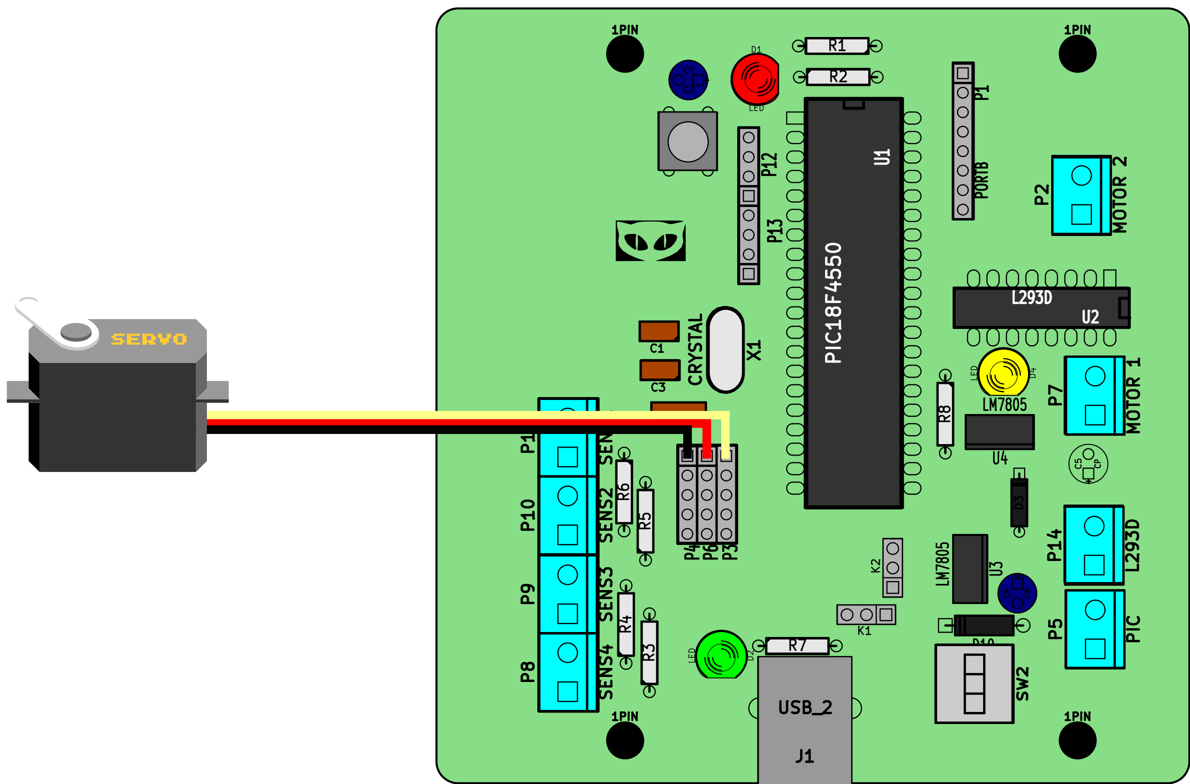
Los servomotores necesitan voltaje para funcionar, la placa NP05 provee 5 volts para su funcionamiento. Sin embargo, si se inserta el conector del servo al revés (es un conector de 3 pines, donde el blanco suele ser la señal, rojo voltaje y negro masa) se puede quemar el servo. Observar el diagrama siguiente para ver la forma correcta de instalar un servo en la placa.



Pines de conexión para servos

4.3. Conectando un servomotor

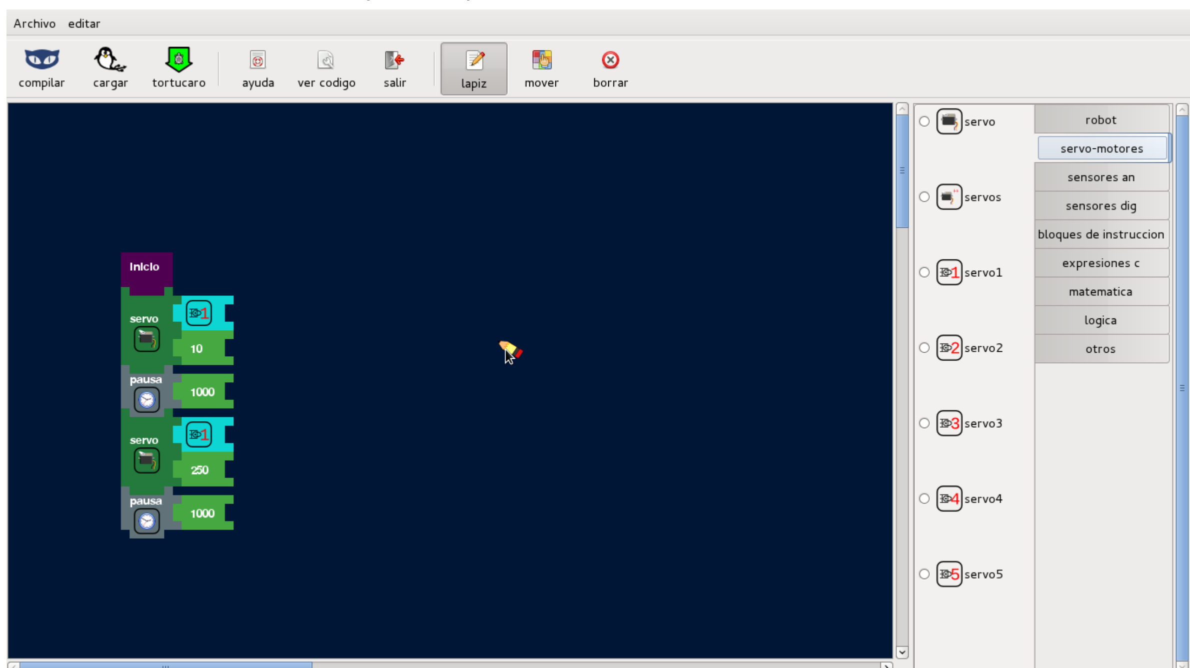
Para conectar un servo a la placa NP05 hay que seguir el diagrama siguiente. Generalmente el cable más claro es el que envía la señal de PWM (en los servos FUTABA S3003 es blanco), el del centro es el voltaje positivo (rojo) y el de la otra punta es la masa (negro). Pueden variar entre fabricantes los códigos de colores, pero en general se respeta el orden de los cables (positivo al medio).



Esquema de conexión de un servo.

En el diagrama anterior se ve cómo el cable de señal PWM tiene que estar en el pin que está más cerca del micro controlador, el del medio es el voltaje positivo y el más alejado del micro es la masa. En ese esquema el servo está conectado al pin 1 de la placa, de ahí hacia abajo son los pines 2,3,4,5.

Para una primera prueba de conexión de un servo, primero abramos el archivo en **`ejemplos/hola_mundo_servos.icr2`** y lo compilemos.



Luego de cargar el firmware, la placa NP05 se activará (a los 5 segundos) y el servo, si está correctamente instalado y andando, comenzará a moverse para un lado y para el otro.

Uso avanzado de `icaro-bloques 1.0`

5.1. Crear componentes nuevos

test test

Apéndice A. Revision History

Revisión 0-0	**Wed Mar 28 2012**	**Valentin Basel** *valentinbasel@gmail.com*
creación inicial con PUBLICAN		

Índice

R